

# General Algorithms for Testing the Ambiguity of Finite Automata

Cyril Allauzen<sup>1,\*</sup>, Mehryar Mohri<sup>1,2</sup>, and Ashish Rastogi<sup>1</sup>

<sup>1</sup> Courant Institute of Mathematical Sciences,  
251 Mercer Street, New York, NY 10012.

<sup>2</sup> Google Research,  
76 Ninth Avenue, New York, NY 10011.

**Abstract.** This paper presents efficient algorithms for testing the finite, polynomial, and exponential ambiguity of finite automata with  $\epsilon$ -transitions. It gives an algorithm for testing the exponential ambiguity of an automaton  $A$  in time  $O(|A|_E^2)$ , and finite or polynomial ambiguity in time  $O(|A|_E^3)$ . These complexities significantly improve over the previous best complexities given for the same problem. Furthermore, the algorithms presented are simple and are based on a general algorithm for the composition or intersection of automata. We also give an algorithm to determine the degree of polynomial ambiguity of a finite automaton  $A$  that is polynomially ambiguous in time  $O(|A|_E^3)$ . Finally, we present an application of our algorithms to an approximate computation of the entropy of a probabilistic automaton.

## 1 Introduction

The question of the ambiguity of finite automata arises in a variety of contexts. In some cases, the application of an algorithm requires an input automaton to be finitely ambiguous, in others the convergence of a bound or guarantee relies on that finite ambiguity or the asymptotic rate of the increase of ambiguity as a function of the string length. Thus, in all these cases, one needs an algorithm to test the ambiguity, either to determine if it is finite, or to estimate its asymptotic rate of increase.

The problem of testing ambiguity has been extensively analyzed in the past. The problem of determining the degree of ambiguity of an automaton with finite ambiguity was shown to be PSPACE-complete. However, testing finite ambiguity can be done in polynomial time using a characterization of polynomial and exponential ambiguity given by [6, 5, 9, 4, 11]. The most efficient algorithms for testing polynomial and exponential ambiguity, and thereby testing finite ambiguity were presented by [10, 12]. The algorithms presented in [12] assume the input automaton to be  $\epsilon$ -free, but they are extended to the case where the automaton has  $\epsilon$ -transitions in [10]. In the presence of  $\epsilon$ -transitions, the complexity of the algorithms given by [10] is  $O((|A|_E + |A|_Q^2)^2)$  for testing the exponential ambiguity of an automaton  $A$  and  $O((|A|_E + |A|_Q^2)^3)$  for testing polynomial ambiguity, where  $|A|_E$  stands for the number of transitions and  $|A|_Q$  the number of states of  $A$ .

---

\* This author's new address is: Google Research, 76 Ninth Avenue, New York, NY 10011.

This paper presents significantly more efficient algorithms for testing finite, polynomial, and exponential ambiguity for the general case of automata with  $\epsilon$ -transitions. It gives an algorithm for testing the exponential ambiguity of an automaton  $A$  in time  $O(|A|_E^2)$ , and finite or polynomial ambiguity in time  $O(|A|_E^3)$ . The main idea behind our algorithms is to make use of the composition or intersection of finite automata with  $\epsilon$ -transitions [8, 7]. The  $\epsilon$ -filter used in these algorithms crucially helps in the analysis and test of the ambiguity. We also give an algorithm to determine the degree of polynomial ambiguity of a finite automaton  $A$  that is polynomially ambiguous in time  $O(|A|_E^3)$ . Finally, we present an application of our algorithms to an approximate computation of the entropy of a probabilistic automaton.

The remainder of the paper is organized as follows. Section 2 presents general automata and ambiguity definitions. In Section 3 we give a brief description of existing characterizations for the ambiguity of automata and extend them to the case of automata with  $\epsilon$ -transitions. In Section 4 we present our algorithms for testing the finite, polynomial, and exponential ambiguity, and the proof of their correctness. Section 5 details the relevance of these algorithms to the approximation of the entropy of probabilistic automata.

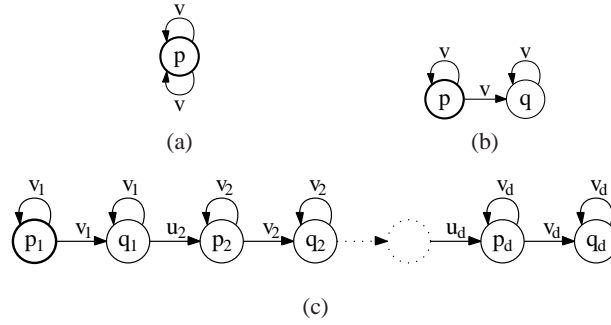
## 2 Preliminaries

**Definition 1.** A finite automaton  $A$  is a 5-tuple  $(\Sigma, Q, E, I, F)$  where:  $\Sigma$  is a finite alphabet;  $Q$  is a finite set of states;  $I \subseteq Q$  the set of initial states;  $F \subseteq Q$  the set of final states; and  $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$  a finite set of transitions, where  $\epsilon$  denotes the empty string.

We denote by  $|A|_Q$  the number of states, by  $|A|_E$  the number of transitions and by  $|A| = |A|_E + |A|_Q$  the size of an automaton  $A$ . Given a state  $q \in Q$ ,  $E[q]$  denotes the set of transitions leaving  $q$ . For two subsets  $R \subseteq Q$  and  $R' \subseteq Q$ , we denote by  $P(R, x, R')$  the set of all paths from a state  $q \in R$  to a state  $q' \in R'$  labeled with  $x \in \Sigma^*$ . We also denote by  $p[\pi]$  the origin state, by  $n[\pi]$  the destination state, and by  $i[\pi] \in \Sigma^*$  the label of a path  $\pi$ .

A string  $x \in \Sigma^*$  is accepted by  $A$  if it labels a successful path, *i.e.* a path from an initial state to a final state. A finite automaton  $A$  is *trim* if every state of  $A$  belongs to a successful path.  $A$  is *unambiguous* if for any string  $x \in \Sigma^*$  there is at most one successful path labeled by  $x$  in  $A$ , otherwise,  $A$  is said *ambiguous*. The *degree of ambiguity* of a string  $x$  in  $A$ , denoted by  $\text{da}(A, x)$ , is the number of successful paths in  $A$  labeled by  $x$ . Note that if  $A$  contains an  $\epsilon$ -cycle, there exist  $x \in \Sigma^*$  such that  $\text{da}(A, x) = \infty$ . Using a depth-first search restricted to  $\epsilon$ -transitions, it can be decided in linear time whether  $A$  has  $\epsilon$ -cycles. Thus, in the following, we will assume without loss of generality that  $A$  is  $\epsilon$ -cycle free.

The *degree of ambiguity* of  $A$  is defined as  $\text{da}(A) = \sup_{x \in \Sigma^*} \text{da}(A, x)$ .  $A$  is said *finitely ambiguous* if  $\text{da}(A) < \infty$  and *infinitely ambiguous* if  $\text{da}(A) = \infty$ .  $A$  is said *polynomially ambiguous* if there exists a polynomial  $h$  in  $\mathbb{N}[X]$  such that  $\text{da}(A, x) \leq h(|x|)$  for all  $x \in \Sigma^*$ . The minimal degree of such a polynomial is called the *degree of polynomial ambiguity* of  $A$ , denoted by  $\text{dpa}(A)$ . By definition,  $\text{dpa}(A) = 0$  iff  $A$  is



**Fig. 1.** Illustration of the (a) (EDA), (b) (IDA) and (c) ( $\text{IDA}_d$ ) properties.

finitely ambiguous. When  $A$  is infinitely ambiguous but not polynomially ambiguous, we say that  $A$  is *exponentially ambiguous* and that  $\text{dpa}(A) = \infty$ .

### 3 Characterization of infinite ambiguity

The characterization and test of finite, polynomial, and exponential ambiguity of finite automata without  $\epsilon$ -transitions are based on the following fundamental properties. [6, 5, 9, 4, 11, 10, 12].

**Definition 2.** *The following are three key properties for the characterization of the ambiguity of an automata  $A$ .*

- (a) (EDA): *There exists a state  $q$  with at least two distinct cycles labeled by some  $v \in \Sigma^*$  (Figure 1(a)).*
- (b) (IDA): *There exist two distinct states  $p$  and  $q$  with paths labeled with  $v$  from  $p$  to  $p$ ,  $p$  to  $q$ , and  $q$  to  $q$ , for some  $v \in \Sigma^*$  (Figure 1(b)).*
- (c) ( $\text{IDA}_d$ ): *There exist  $2d$  states  $p_1, \dots, p_d, q_1, \dots, q_d$  in  $A$  and  $2d-1$  strings  $v_1, \dots, v_d$  and  $u_2, \dots, u_d$  in  $\Sigma^*$  such that for all  $1 \leq i \leq d$ ,  $p_i \neq q_i$  and  $P(p_i, v_i, p_i)$ ,  $P(p_i, v_i, q_i)$  and  $P(q_i, v_i, q_i)$  are non-empty and for all  $2 \leq i \leq d$ ,  $P(q_{i-1}, u_i, p_i)$  is non-empty (Figure 1(c)).*

Observe that (EDA) implies (IDA). Assuming (EDA), let  $e$  and  $e'$  be the first transitions that differ in the two cycles at state  $q$ , then we must have  $n[e] \neq n[e']$  since the definition 1 disallows multiple transitions between the same two states with the same label. Thus, (IDA) holds for the pair  $(n[e], n[e'])$ .

In the  $\epsilon$ -free case, it was shown that a trim automaton  $A$  satisfies (IDA) iff  $A$  is infinitely ambiguous [11, 12], that  $A$  satisfies (EDA) iff  $A$  is exponentially ambiguous [4], and that  $A$  satisfies ( $\text{IDA}_d$ ) iff  $\text{dpa}(A) \geq d$  [10, 12]. These characterizations can be straightforwardly extended to the case of automata with  $\epsilon$ -transitions in the following proposition.

**Proposition 1.** *Let  $A$  be a trim  $\epsilon$ -cycle free finite automaton.*

- (i)  $A$  is infinitely ambiguous iff  $A$  satisfies (IDA).
- (ii)  $A$  is exponentially ambiguous iff  $A$  satisfies (EDA).
- (iii)  $\text{dpa}(A) \geq d$  iff  $A$  satisfies  $(\text{IDA}_d)$ .

*Proof.* The proof is by induction on the number of  $\epsilon$ -transitions in  $A$ . If  $A$  does not have any  $\epsilon$ -transitions, then the proposition holds as shown in [11, 12] for (i), [4] for (ii) and [12] for (iii).

Assume now that  $A$  has  $n + 1$   $\epsilon$ -transitions,  $n \geq 0$ , and that the statement of the proposition holds for all automata with  $n$   $\epsilon$ -transitions. Select an  $\epsilon$ -transition  $e_0$  in  $A$ , and let  $A'$  be the finite automaton obtained after application of  $\epsilon$ -removal to  $A$  limited to transition  $e_0$ .  $A'$  is obtained by deleting  $e_0$  from  $A$  and by adding a transition  $(p[e_0], l[e], n[e])$  for every transition  $e \in E[n[e_0]]$ . It is clear that  $A$  and  $A'$  are equivalent and that there is a label-preserving bijection between the paths in  $A$  and  $A'$ . Thus, (a)  $A$  satisfies (IDA) (resp. (EDA),  $(\text{IDA}_d)$ ) iff  $A'$  satisfies (IDA) (resp. (EDA),  $(\text{IDA}_d)$ ) and (b) for all  $x \in \Sigma^*$ ,  $\text{da}(A, x) = \text{da}(A', x)$ . By induction, proposition 1 holds for  $A'$  and thus, it follows from (a) and (b) that proposition 1 also holds for  $A$ .  $\square$

These characterizations have been used in [10, 12] to design algorithms for testing infinite, polynomial, and exponential ambiguity, and for computing the degree of polynomial ambiguity in the  $\epsilon$ -free case.

**Theorem 1 ([10, 12]).** *Let  $A$  be a trim  $\epsilon$ -free finite automaton.*

1. *It is decidable in time  $O(|A|_E^3)$  whether  $A$  is infinitely ambiguous.*
2. *It is decidable in time  $O(|A|_E^2)$  whether  $A$  is exponentially ambiguous.*
3. *The degree of polynomial ambiguity of  $A$ ,  $\text{dpa}(A)$ , can be computed in  $O(|A|_E^3)$ .*

The first result of theorem 1 has also been generalized by [10] to the case of automata with  $\epsilon$ -transitions but with a significantly worse complexity.

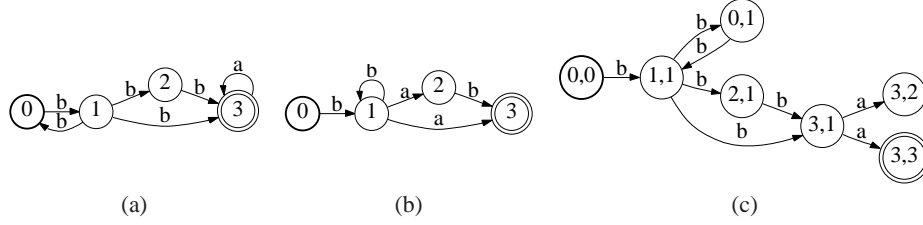
**Theorem 2 ([10]).** *Let  $A$  be a trim  $\epsilon$ -cycle free finite automaton. It is decidable in time  $O((|A|_E + |A|_Q^2)^3)$  whether  $A$  is infinitely ambiguous.*

The main idea used in [10] is to defined from  $A$  an  $\epsilon$ -free automaton  $A'$  such that  $A$  is infinitely ambiguous iff  $A'$  is infinitely ambiguous. However, the number of transitions of  $A'$  is  $|A|_E + |A|_Q^2$ . This explains why the complexity in the  $\epsilon$ -transition case is significantly worse than in the  $\epsilon$ -free case. A similar approach can be used straightforwardly to test the exponential ambiguity of  $A$  with complexity  $O((|A|_E + |A|_Q^2)^2)$  and to compute  $\text{dpa}(A)$  when  $A$  is polynomially ambiguous with complexity  $O((|A|_E + |A|_Q^2)^3)$ .

Note that we give here tighter estimates of the complexity of the algorithms of [10, 12] where the authors gave complexities using the loose inequality:  $|A|_E \leq |\Sigma| \cdot |A|_Q^2$ .

## 4 Algorithms

Our algorithms for testing ambiguity are based on a general algorithm for the composition or intersection of automata, which we describe in the following section both to be self-contained, and to give a proof of the correctness of the  $\epsilon$ -filter which we have not presented in earlier publications.



**Fig. 2.** Example of finite automaton intersection. (a) Finite automata  $A_1$  and (b)  $A_2$ . (c) Result of the intersection of  $A_1$  and  $A_2$ .

#### 4.1 Intersection of finite automata

The intersection of finite automata is a special case of the general composition algorithm for weighted transducers [8, 7]. States in the intersection  $A_1 \cap A_2$  of two finite automata  $A_1$  and  $A_2$  are identified with pairs of a state of  $A_1$  and a state of  $A_2$ . Leaving aside  $\epsilon$ -transitions, the following rule specifies how to compute a transition of  $A_1 \cap A_2$  from appropriate transitions of  $A_1$  and  $A_2$ :

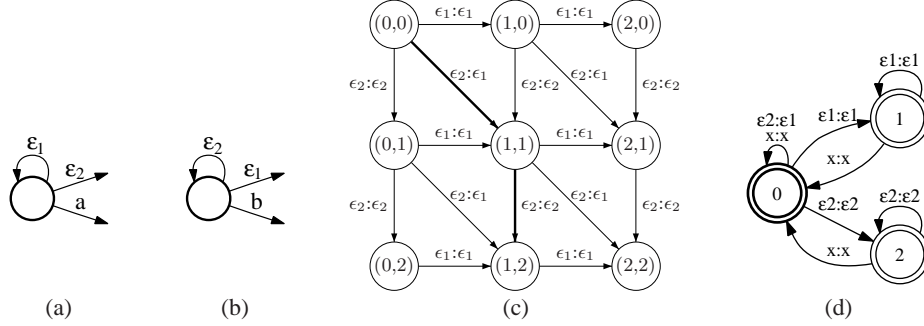
$$(q_1, a, q'_1) \text{ and } (q_2, a, q'_2) \implies ((q_1, q'_1), a, (q_2, q'_2)). \quad (1)$$

Figure 2 illustrates the algorithm. A state  $(q_1, q_2)$  is initial (resp. final) when  $q_1$  and  $q_2$  are initial (resp. final). In the worst case, all transitions of  $A_1$  leaving a state  $q_1$  match all those of  $A_2$  leaving state  $q_2$ , thus the space and time complexity of composition is quadratic:  $O(|A_1||A_2|)$ , or  $O(|A_1|_E|A_2|_E)$  when  $A_1$  and  $A_2$  are trim.

**Epsilon filtering** A straightforward generalization of the  $\epsilon$ -free case would generate redundant  $\epsilon$ -paths. This is a crucial issue in the more general case of the intersection of weighted automata over a non-idempotent semiring, since it would lead to an incorrect result. The weight of two matching  $\epsilon$ -paths of the original automata would then be counted as many times as the number of redundant  $\epsilon$ -paths generated in the result, instead of one. It is also a crucial problem in the unweighted case that we are considering since redundant  $\epsilon$ -paths can affect the test of infinite ambiguity, as we shall see in the next section. A critical component of the composition algorithm of [8, 7] consists however of precisely coping with this problem using a method called *epsilon filtering*.

Figure 3(c) illustrates the problem just mentioned. To match  $\epsilon$ -paths leaving  $q_1$  and those leaving  $q_2$ , a generalization of the  $\epsilon$ -free intersection can make the following moves: (1) first move forward on an  $\epsilon$ -transition of  $q_1$ , or even a  $\epsilon$ -path, and stay at the same state  $q_2$  in  $A_2$ , with the hope of later finding a transition whose label is some label  $a \neq \epsilon$  matching a transition of  $q_2$  with the same label; (2) proceed similarly by following an  $\epsilon$ -transition or  $\epsilon$ -path leaving  $q_2$  while staying at the same state  $q_1$  in  $A_1$ ; or, (3) match an  $\epsilon$ -transition of  $q_1$  with an  $\epsilon$ -transition of  $q_2$ .

Let us rename existing  $\epsilon$ -labels of  $A_1$  as  $\epsilon_2$ , and existing  $\epsilon$ -labels of  $A_2$  as  $\epsilon_1$ , and let us augment  $A_1$  with a self-loop labeled with  $\epsilon_1$  at all states and similarly, augment  $A_2$  with a self-loop labeled with  $\epsilon_2$  at all states, as illustrated by Figures 3(a) and (b). These



**Fig. 3.** Marking of automata, redundant paths and filter. (a)  $\tilde{A}_1$ : self-loop labeled with  $\epsilon_1$  added at all states of  $A_1$ , regular  $\epsilon$ s renamed to  $\epsilon_2$ . (b)  $\tilde{A}_2$ : self-loop labeled with  $\epsilon_2$  added at all states of  $A_2$ , regular  $\epsilon$ s renamed to  $\epsilon_1$ . (c) Redundant  $\epsilon$ -paths: a straightforward generalization of the  $\epsilon$ -free case could generate all the paths from  $(0, 0)$  to  $(2, 2)$  for example, even when composing just two simple transducers. (d) Filter transducer  $M$  allowing a unique  $\epsilon$ -path.

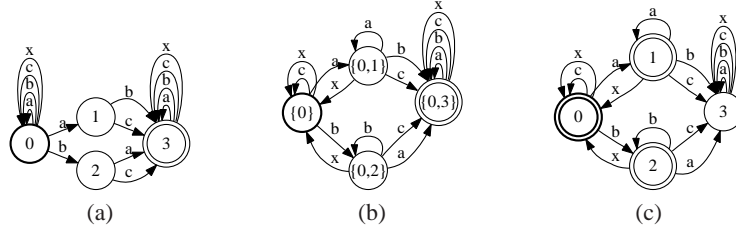
self-loops correspond to staying at the same state in that machine while consuming an  $\epsilon$ -label of the other transition. The three moves just described now correspond to the matches (1)  $(\epsilon_2:\epsilon_2)$ , (2)  $(\epsilon_1:\epsilon_1)$ , and (3)  $(\epsilon_2:\epsilon_1)$ . The grid of Figure 3(c) shows all the possible  $\epsilon$ -paths between intersection states. We will denote by  $\tilde{A}_1$  and  $\tilde{A}_2$  the automata obtained after application of these changes.

For the result of intersection not to be redundant, between any two of these states, all but one path must be disallowed. There are many possible ways of selecting that path. One natural way is to select the shortest path with the diagonal transitions ( $\epsilon$ -matching transitions) taken first. Figure 3(c) illustrates in boldface the path just described from state  $(0, 0)$  to state  $(1, 2)$ . Remarkably, this filtering mechanism itself can be encoded as a finite-state transducer such as the transducer  $M$  of Figure 3(d). We denote by  $(p, q) \preceq (r, s)$  to indicate that  $(r, s)$  can be reached from  $(p, q)$  in the grid.

**Proposition 2.** *Let  $M$  be the transducer of Figure 3(d).  $M$  allows a unique path between any two states  $(p, q)$  and  $(r, s)$ , with  $(p, q) \preceq (r, s)$ .*

*Proof.* Let  $a$  denote  $(\epsilon_1:\epsilon_1)$ ,  $b$  denote  $(\epsilon_2:\epsilon_2)$ ,  $c$  denote  $(\epsilon_2:\epsilon_1)$ , and let  $x$  stand for any  $(x:x)$ , with  $x \in \Sigma$ . The following sequences must be disallowed by a shortest-path filter with matching transitions first:  $ab, ba, ac, bc$ . This is because, from any state, instead of the moves  $ab$  or  $ba$ , the matching or diagonal transition  $c$  can be taken. Similarly, instead of  $ac$  or  $bc$ ,  $ca$  and  $cb$  can be taken for an earlier match. Conversely, it is clear from the grid or an immediate recursion that a filter disallowing these sequences accepts a unique path between two connected states of the grid.

Let  $L$  be the set of sequences over  $\sigma = \{a, b, c, x\}$  that contain one of the disallowed sequence just mentioned as a substring that is  $L = \sigma^*(ab + ba + ac + bc)\sigma^*$ . Then  $\bar{L}$  represents exactly the set of paths allowed by that filter and is thus a regular language. Let  $A$  be an automaton representing  $L$  (Figure 4(a)). An automaton representing  $\bar{L}$  can



**Fig. 4.** (a) Finite automaton  $A$  representing the set of disallowed sequences. (b) Automaton  $B$ , result of the determinization of  $A$ . Subsets are indicated at each state. (c) Automaton  $C$  obtained from  $B$  by complementation, state 3 is not coaccessible.

be constructed from  $A$  by determinization and complementation (Figures 4(a)-(c)). The resulting automaton  $C$  is equivalent to the transducer  $M$  after removal of the state 3, which does not admit a path to a final state.  $\square$

Thus, to intersect two finite automata  $A_1$  and  $A_2$  with  $\epsilon$ -transitions, it suffices to compute  $\tilde{A}_1 \circ M \circ \tilde{A}_2$ , using the  $\epsilon$ -free rules of intersection or composition.

**Theorem 3.** *Let  $A_1$  and  $A_2$  be two finite automata with  $\epsilon$ -transitions. To each pair  $(\pi_1, \pi_2)$  of successful paths in  $A_1$  and  $A_2$  sharing the same input label  $x \in \Sigma^*$  corresponds a unique successful path  $\pi$  in  $A_1 \cap A_2$  labeled by  $x$ .*

*Proof.* This follows straightforwardly from proposition 2.  $\square$

## 4.2 Testing for infinite ambiguity

We start with a test of the exponential ambiguity of  $A$ . The key is that the (EDA) property translates into a very simple property for  $A^2 = A \cap A$ .

**Lemma 1.** *Let  $A$  be a trim  $\epsilon$ -cycle free finite automaton.  $A$  satisfies (EDA) iff there exists a strongly connected component of  $A^2 = A \cap A$  that contains two states of the form  $(p, p)$  and  $(q, q')$ , where  $p, q$  and  $q'$  are states of  $A$  with  $q \neq q'$ .*

*Proof.* Assume that  $A$  satisfies (EDA). There exist a state  $p$  and a string  $v$  such that there are two distinct cycles  $c_1$  and  $c_2$  labeled by  $v$  at  $p$ . Let  $e_1$  and  $e_2$  be the first edges that differ in  $c_1$  and  $c_2$ . We can then write  $c_1 = \pi e_1 \pi_1$  and  $c_2 = \pi e_2 \pi_2$ . If  $e_1$  and  $e_2$  share the same label, let  $\pi'_1 = \pi e_1$ ,  $\pi'_2 = \pi e_2$ ,  $\pi''_1 = \pi_1$  and  $\pi''_2 = \pi_2$ . If  $e_1$  and  $e_2$  do not share the same label, exactly one of them must be an  $\epsilon$ -transition. By symmetry, we can assume without loss of generality that  $e_1$  is the  $\epsilon$ -transition. Let  $\pi'_1 = \pi e_1$ ,  $\pi'_2 = \pi$ ,  $\pi''_1 = \pi_1$  and  $\pi''_2 = \epsilon_2 \pi_2$ . In both cases, let  $q = n[\pi'_1] = p[\pi''_1]$  and  $q' = n[\pi'_2] = p[\pi''_2]$ . Observe that  $q \neq q'$ . Since  $i[\pi'_1] = i[\pi'_2]$ ,  $\pi'_1$  and  $\pi'_2$  are matched by intersection resulting in a path in  $A^2$  from  $(p, p)$  to  $(q, q')$ . Similarly, since  $i[\pi''_1] = i[\pi''_2]$ ,  $\pi''_1$  and  $\pi''_2$  are matched by intersection resulting in a path from  $(q, q')$  to  $(p, p)$ . Thus,  $(p, p)$  and  $(q, q')$  are in the same strongly connected component of  $A^2$ .



Conversely, assume that there exist states  $p, q$  and  $q'$  in  $A$  such that  $q \neq q'$  and that  $(p, p)$  and  $(q, q')$  are in the same strongly connected component of  $A^2$ . Let  $c$  be a cycle in  $(p, p)$  going through  $(q, q')$ , it has been obtained by matching two cycles  $c_1$  and  $c_2$ . If  $c_1$  were equal to  $c_2$ , intersection would match these two paths creating a path  $c'$  along which all the states would be of the form  $(r, r)$ , and since  $A$  is trim this would contradict Theorem 3. Thus,  $c_1$  and  $c_2$  are distinct and (EDA) holds.  $\square$

Lemma 1 leads to a straightforward algorithm for testing exponential ambiguity.

**Theorem 4.** *Let  $A$  be a trim  $\epsilon$ -cycle free finite automaton. It is decidable in time  $O(|A|_E^2)$  whether  $A$  is exponentially ambiguous.*

*Proof.* The algorithm proceeds as follows. We compute  $A^2$  and, using a depth-first search of  $A^2$ , trim it and compute its strongly connected components. It follows from Lemma 1 that  $A$  is exponentially ambiguous iff there is a strongly connected component that contains two states of the form  $(p, p)$  and  $(q, q')$  with  $q \neq q'$ . Finding such a strongly connected component can be done in time linear in the size of  $A^2$ , i.e. in  $O(|A|_E^2)$  since  $A$  and  $A^2$  are trim. Thus, the complexity of the algorithm is in  $O(|A|_E^2)$ .  $\square$

Testing the (IDA) property requires finding three paths sharing the same label in  $A$ . This can be done in a natural way using the automaton  $A^3 = A \cap A \cap A$ , as shown below.

**Lemma 2.** *Let  $A$  be a trim  $\epsilon$ -cycle free finite automaton.  $A$  satisfies (IDA) iff there exist two distinct states  $p$  and  $q$  in  $A$  with a non- $\epsilon$  path in  $A^3 = A \cap A \cap A$  from state  $(p, p, q)$  to state  $(p, q, q)$ .*

*Proof.* Assume that  $A$  satisfies (IDA). Then, there exists a string  $v \in \Sigma^*$  with three paths  $\pi_1 \in P(p, v, p)$ ,  $\pi_2 \in P(p, v, q)$  and  $\pi_3 \in P(q, v, p)$ . Since these three paths share the same label  $v$ , they are matched by intersection resulting in a path  $\pi$  in  $A^3$  labeled with  $v$  from  $(p[\pi_1], p[\pi_2], p[\pi_3]) = (p, p, q)$  to  $(n[\pi_1], n[\pi_2], n[\pi_3]) = (p, q, q)$ .

Conversely, if there is a non- $\epsilon$  path  $\pi$  from  $(p, p, q)$  to  $(p, q, q)$  in  $A^3$ , it has been obtained by matching three paths  $\pi_1, \pi_2$  and  $\pi_3$  in  $A$  with the same input  $v = i[\pi] \neq \epsilon$ . Thus, (IDA) holds.  $\square$

Finally, Theorem 4 and Lemma 2 can be combined to yield the following result.

**Theorem 5.** *Let  $A$  be a trim  $\epsilon$ -cycle free finite automaton. It is decidable in time  $O(|A|_E^3)$  whether  $A$  is finitely, polynomially, or exponentially ambiguous.*

*Proof.* First, Theorem 4 can be used to test whether  $A$  is exponentially ambiguous by computing  $A^2$ . The complexity of this step is  $O(|A|_E^2)$ .

If  $A$  is not exponentially ambiguous, we proceed by computing and trimming  $A^3$  and then testing whether  $A^3$  verifies the property described in lemma 2. This is done by considering the automaton  $B$  on the alphabet  $\Sigma' = \Sigma \cup \{\#\}$  obtained from  $A^3$  by adding a transition labeled by  $\#$  from state  $(p, q, q)$  to state  $(p, p, q)$  for every pair  $(p, q)$  of states in  $A$  such that  $p \neq q$ . It follows that  $A^3$  verifies the condition in lemma 2 iff there is a cycle in  $B$  containing both a transition labeled by  $\#$  and a transition labeled



by a symbol in  $\Sigma$ . This property can be checked straightforwardly using a depth-first search of  $B$  to compute its strongly connected components. If a strongly connected component of  $B$  is found that contains both a transition labeled with  $\#$  and a transition labeled by a symbol in  $\Sigma$ ,  $A$  verifies (IDA) but not (EDA) and thus  $A$  is polynomially ambiguous. Otherwise,  $A$  is finitely ambiguous. The complexity of this step is linear in the size of  $B$ :  $O(|B|_E) = O(|A_E|^3 + |A_Q|^2) = O(|A_E|^3)$  since  $A$  and  $B$  are trim.

The total complexity of the algorithm is  $O(|A|_E^2 + |A|_E^3) = O(|A|_E^3)$ .

When  $A$  is polynomially ambiguous, we can derive from the algorithm just described one that computes  $\text{dpa}(A)$ .

**Theorem 6.** *Let  $A$  be a trim  $\epsilon$ -cycle free finite automaton. If  $A$  is polynomially ambiguous,  $\text{dpa}(A)$  can be computed in time  $O(|A|_E^3)$ .*

*Proof.* We first compute  $A^3$  and use the algorithm of theorem 5 to test whether  $A$  is polynomially ambiguous and to compute all the pairs  $(p, q)$  that verify the condition of Lemma 2. This step has complexity  $O(|A|_E^3)$ .

We then compute the component graph  $G$  of  $A$ , and for each pair  $(p, q)$  found in the previous step, we add a transition labeled with  $\#$  from the strongly connected component of  $p$  to the one of  $q$ . If there is a path in that graph containing  $d$  edges labeled by  $\#$ , then  $A$  verifies  $(\text{IDA}_d)$ . Thus,  $\text{dpa}(A)$  is the maximum number of edges marked by  $\#$  that can be found along a path in  $G$ . Since  $G$  is acyclic, this number can be computed in linear time in the size of  $G$ , i.e. in  $O(|A|_Q^2)$ . Thus, the overall complexity of the algorithm is  $O(|A|_E^3)$ .  $\square$

## 5 Application to the Approximation of Entropy

In this section, we describe an application in which determining the degree of ambiguity of a *probabilistic* automaton helps estimate the quality of an approximation of its entropy.

Weighted automata are automata in which each transition carries some weight in addition to the usual alphabet symbol. The weights are elements of a semiring, that is a ring that may lack negation. The following is a more formal definition.

**Definition 3.** A weighted automaton  $A$  over a semiring  $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$  is a 7-tuple  $(\Sigma, Q, I, F, E, \lambda, \rho)$  where:  $\Sigma$  is the finite alphabet of the automaton,  $Q$  is a finite set of states,  $I \subseteq Q$  the set of initial states,  $F \subseteq Q$  the set of final states,  $E \subseteq Q \times \Sigma \cup \{\epsilon\} \times \mathbb{K} \times Q$  a finite set of transitions,  $\lambda : I \rightarrow \mathbb{K}$  the initial weight function mapping  $I$  to  $\mathbb{K}$ , and  $\rho : F \rightarrow \mathbb{K}$  the final weight function mapping  $F$  to  $\mathbb{K}$ .

Given a transition  $e \in E$ , we denote by  $w[e]$  its weight. We extend the weight function  $w$  to paths by defining the weight of a path as the  $\otimes$ -product of the weights of its constituent transitions:  $w[\pi] = w[e_1] \otimes \dots \otimes w[e_k]$ . The weight associated by a weighted automaton  $A$  to an input string  $x \in \Sigma^*$  is defined by:

$$\llbracket A \rrbracket(x) = \bigoplus_{\pi \in P(I, x, F)} \lambda[p[\pi]] \otimes w[\pi] \otimes \rho[n[\pi]]. \quad (2)$$

The entropy  $H(A)$  of a probabilistic automaton  $A$  is defined as:

$$H(A) = - \sum_{x \in \Sigma^*} \llbracket A \rrbracket(x) \log(\llbracket A \rrbracket(x)). \quad (3)$$

Let  $\mathbb{K}$  denote  $(\mathbb{R} \cup \{+\infty, -\infty\}) \times (\mathbb{R} \cup \{+\infty, -\infty\})$ . The system  $(\mathbb{K}, \oplus, \otimes, (0, 0), (1, 0))$  where  $\oplus$  and  $\otimes$  are defined as follows defines a commutative semiring called the *entropy semiring* [2]. For any two pairs  $(x_1, y_1)$  and  $(x_2, y_2)$  in  $\mathbb{K}$ ,

$$(x_1, y_1) \oplus (x_2, y_2) = (x_1 + x_2, y_1 + y_2) \quad (4)$$

$$(x_1, y_1) \otimes (x_2, y_2) = (x_1 x_2, x_1 y_2 + x_2 y_1). \quad (5)$$

In [2], the authors show that a generalized shortest-distance algorithm over this semiring correctly computes the entropy of an unambiguous probabilistic automaton  $A$ . The algorithm starts by mapping the weight of each transition to a pair where the first element is the probability and the second the entropy:  $w[e] \mapsto (w[e], -w[e] \log w[e])$ . The algorithm then proceeds by computing the generalized shortest-distance under the *entropy semiring*, which computes the  $\oplus$ -sum of the weights of all accepting paths in  $A$ .

In this section, we show that the same shortest-distance algorithm yields an approximation of the entropy of an ambiguous probabilistic automaton  $A$ , where the approximation quality is a function of the degree of polynomial ambiguity,  $\text{dpa}(A)$ . Our proofs make use of the standard log-sum inequality [3], a special case of Jensen's inequality, which holds for any positive reals  $a_1, \dots, a_k$ , and  $b_1, \dots, b_k$ :

$$\sum_{i=1}^k a_i \log \frac{a_i}{b_i} \geq \left( \sum_{i=1}^k a_i \right) \log \frac{\sum_{i=1}^k a_i}{\sum_{i=1}^k b_i}. \quad (6)$$

**Lemma 3.** *Let  $A$  be a probabilistic automaton and let  $x \in \Sigma^+$  be a string accepted by  $A$  on  $k$  paths  $\pi_1, \dots, \pi_k$ . Let  $w(\pi_i)$  be the probability of path  $\pi_i$ . Clearly,  $\llbracket A \rrbracket(x) = \sum_{i=1}^k w(\pi_i)$ . Then,*

$$\sum_{i=1}^k w(\pi_i) \log w(\pi_i) \geq \llbracket A \rrbracket(x) (\log \llbracket A \rrbracket(x) - \log k). \quad (7)$$

*Proof.* The result follows straightforwardly from the log-sum inequality, with  $a_i = w(\pi_i)$  and  $b_i = 1$ :

$$\sum_{i=1}^k w(\pi_i) \log w(\pi_i) \geq \left( \sum_{i=1}^k w(\pi_i) \right) \log \frac{\sum_{i=1}^k w(\pi_i)}{k} = \llbracket A \rrbracket(x) (\log \llbracket A \rrbracket(x) - \log k). \quad (8)$$

□

For a probabilistic automaton  $A$ , let  $S(A)$  be the quantity computed by the generalized shortest-distance algorithm with the entropy semiring. For an unambiguous automaton  $A$ ,  $S(A) = H(A)$  [2].

**Theorem 7.** Let  $A$  be a probabilistic automaton and let  $L$  denote the expected length of strings accepted by  $A$  (i.e.  $L = \sum_{x \in \Sigma^*} |x| \llbracket A \rrbracket(x)$ ). Then,

1. If  $A$  is finitely ambiguous with degree of ambiguity  $k$  (i.e.  $\text{da}(A) = k$  for some  $k \in \mathbb{N}$ ), then  $H(A) \leq S(A) \leq H(A) + \log k$ .
2. If  $A$  is polynomially ambiguous with degree of polynomial ambiguity  $k$  (i.e.  $\text{dpa}(A) = k$  for some  $k \in \mathbb{N}$ ), then  $H(A) \leq S(A) \leq H(A) + k \log L$ .

*Proof.* The lower bound,  $S(A) \geq H(A)$  follows from the observation that for a string  $x$  that is accepted in  $A$  by  $k$  paths  $\pi_1, \dots, \pi_k$ ,

$$\sum_{i=1}^k w(\pi_i) \log(w(\pi_i)) \leq \left( \sum_{i=1}^k w(\pi_i) \right) \log \left( \sum_{i=1}^k w(\pi_i) \right). \quad (9)$$

Since the quantity  $-\sum_{i=1}^k w(\pi_i) \log(w(\pi_i))$  is string  $x$ 's contribution to  $S(A)$  and the quantity  $-(\sum_{i=1}^k w(\pi_i)) \log(\sum_{i=1}^k w(\pi_i))$  its contribution to  $H(A)$ , summing over all accepted strings  $x$ , we obtain  $H(A) \leq S(A)$ .

Assume that  $A$  is finitely ambiguous with degree of ambiguity  $k$ . Let  $x \in \Sigma^*$  be a string that is accepted on  $l_x \leq k$  paths  $\pi_1, \dots, \pi_{l_x}$ . By Lemma 3,

$$\sum_{i=1}^{l_x} w(\pi_i) \log w(\pi_i) \geq \llbracket A \rrbracket(x) (\log \llbracket A \rrbracket(x) - \log l_x) \geq \llbracket A \rrbracket(x) (\log \llbracket A \rrbracket(x) - \log k). \quad (10)$$

Thus,

$$S(A) = - \sum_{x \in \Sigma^*} \sum_{i=1}^{l_x} w(\pi_i) \log w(\pi_i) \leq H(A) + \sum_{x \in \Sigma^*} (\log k) \llbracket A \rrbracket(x) = H(A) + \log k. \quad (11)$$

This proves the first statement of the theorem.

Next, assume that  $A$  is polynomially ambiguous with degree of polynomial ambiguity  $k$ . By Lemma 3,

$$\sum_{i=1}^{l_x} w(\pi_i) \log w(\pi_i) \geq \llbracket A \rrbracket(x) (\log \llbracket A \rrbracket(x) - \log l_x) \geq \llbracket A \rrbracket(x) (\log \llbracket A \rrbracket(x) - \log(|x|^k)). \quad (12)$$

Thus,

$$\begin{aligned} S(A) &\leq H(A) + \sum_{x \in \Sigma^*} k \llbracket A \rrbracket(x) \log |x| = H(A) + k \mathbb{E}_A[\log |x|] \\ &\leq H(A) + k \log \mathbb{E}_A[|x|] = H(A) + k \log L, \quad (\text{by Jensen's inequality}) \end{aligned} \quad (13)$$

which proves the second statement of the theorem.  $\square$

The quality of the approximation of the entropy of a probabilistic automaton  $A$  depends on the expected length  $L$  of an accepted string.  $L$  can be computed efficiently for an arbitrary probabilistic automaton using the *expectation semiring* and the generalized shortest-distance algorithms, using techniques similar to the ones described in [2]. The definition of the expectation semiring is identical to the entropy semiring. The only difference is in the initial step, where the weight of each transition in  $A$  is mapped to a pair of elements. Under the expectation semiring, the mapping is  $w[e] \mapsto (w[e], w[e])$ .

## 6 Conclusion

We presented simple and efficient algorithms for testing the finite, polynomial, or exponential ambiguity of finite automata with  $\epsilon$ -transitions. We conjecture that the running-time complexity of our algorithms is optimal. These algorithms have a variety of applications, in particular to test a pre-condition for the applicability of other automata algorithms. Our application to the approximation of the entropy gives another illustration of the applications of these algorithms.

Our algorithms also illustrate the prominent role played by the general algorithm for the intersection or composition of automata and transducers with  $\epsilon$ -transitions in the design of *testing algorithms*. Composition can be used to devise simple and efficient testing algorithms. We have shown elsewhere how it can be used to test the functionality of a finite-state transducer or to test the twins property for weighted automata and transducers [1].

**Acknowledgments.** The research of Cyril Allauzen and Mehryar Mohri was partially supported by the New York State Office of Science Technology and Academic Research (NYSTAR). This project was also sponsored in part by the Department of the Army Award Number W81XWH-04-1-0307. The U.S. Army Medical Research Acquisition Activity, 820 Chandler Street, Fort Detrick MD 21702-5014 is the awarding and administering acquisition office. The content of this material does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

## References

1. Cyril Allauzen and Mehryar Mohri. Efficient Algorithms for Testing the Twins Property. *Journal of Automata, Languages and Combinatorics*, 8(2):117–144, 2003.
2. Corinna Cortes, Mehryar Mohri, Ashish Rastogi, and Michael Riley. Efficient computation of the relative entropy of probabilistic automata. In *LATIN 2006*, volume 3887 of *Lecture Notes in Computer Science*, pages 323–336. Springer, 2006.
3. Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., New York, 1991.
4. Oscar H. Ibarra and Bala Ravikumar. On sparseness, ambiguity and other decision problems for acceptors and transducers. In *STACS 1986*, volume 210 of *Lecture Notes in Computer Science*, pages 171–179. Springer, 1986.
5. Gérard Jacob. Un algorithme calculant le cardinal, fini ou infini, des demi-groupes de matrices. *Theoretical Computer Science*, 5(2):183–202, 1977.
6. Arnaldo Mandel and Imre Simon. On finite semigroups of matrices. *Theoretical Computer Science*, 5(2):101–111, 1977.
7. Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. Weighted Automata in Text and Speech Processing. In *Proceedings of the 12th biennial European Conference on Artificial Intelligence (ECAI-96)*. John Wiley and Sons, 1996.
8. Fernando Pereira and Michael Riley. *Finite State Language Processing*, chapter Speech Recognition by Composition of Weighted Finite Automata. The MIT Press, 1997.
9. Christophe Reutenauer. Propriétés arithmétiques et topologiques des séries rationnelles en variable non commutative. Thèse troisième cycle, Université Paris VI, 1977.

10. Andreas Weber. Über die Mehrdeutigkeit und Wertigkeit von endlichen, Automaten und Transducern. Dissertation, Goethe-Universität Frankfurt am Main, 1987.
11. Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. In *MFCS 1986*, volume 233 of *Lecture Notes in Computer Science*, pages 620–629. Springer, 1986.
12. Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. *Theoretical Computer Science*, 88(2):325–349, 1991.